Model Development Using Machine Learning Models: Hyperparameter Tuning

September 18, 2023

Hyperparameter Tuning

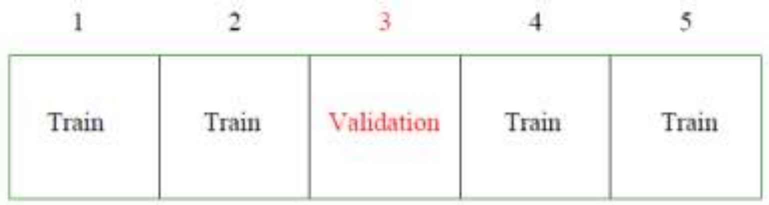
Supervised Machine Learning: Tuning

- **Hyper-parameter tuning** - find the best hyper parameters which gives the most accurate machine learning algorithm.
- HPs are data dependent - need to be tuned to get the best model
  - Simple model structure, small data - simple algorithm
  - complicated model structure with large data - complicated algorithm.
- Tuning - **search** method and a **fitting-evaluation** method.
  - For each HP setting α, fit the model $\hat{f}(x;\alpha)$ and evaluate the model performance; Using the search method to find the hyper-parameter/model that optimizes the model performance.

- It is well-known that a model that minimizes the loss/error on the training data is likely to overfit. To avoid this, the performance is measured on **a separate validation data** (when there is abundant data), or using **cross-validation** (when there is not enough data).

Evaluation Method: Cross Validation

- Cross-validation. The typical **K-fold cross validation** works as follows:
  1. Randomly divide the data into K folds. (Stratification may be needed for imbalanced data)
  2. For each k = 1, …, K
     1. Leave the k-th fold out, build a model using the rest K-1 folds and given hyper parameter α, denote as $\hat{f}^{-k}(x;\alpha)$
     2. Predict on the k-th fold.
  3. After obtaining the cross-validation predictions for the entire data, compute the loss/error

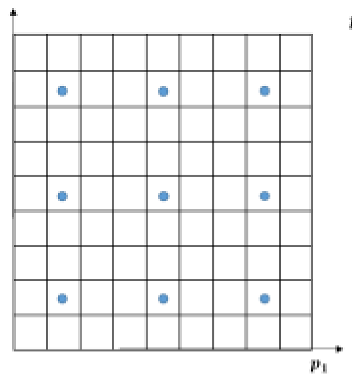


  This is the cross-validation model performance (sometimes, people also compute the performance for each k-th hold-out fold and do average).

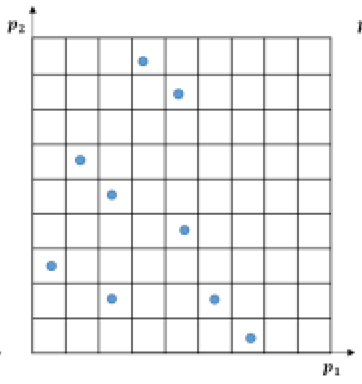
- Typical choices for K are 5 or 10. The case k=N is known as **leave-one-out** cross validation. With larger K, it gets more computationally expensive.

Batch or non-sequential search methods

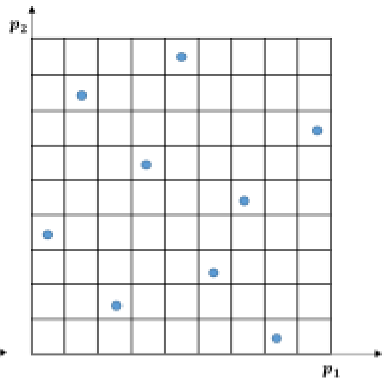
- **Grid search** and **random search :** two most widely-used batch (non-sequential) parameter tuning methods.
- **Grid search** - specify a set of grid points for each parameter and try all combinations in the parameter grid space. It is simple but not efficient:
  - The number of HP combinations increases exponentially; yet many parameters may not matter much.
  - On any single HP dimension, you only have a handful of points, this is risky.
- **Random search**, we randomly sample hyper-parameters from each parameter space.
  - "*Random Search for Hyper-Parameter Optimization*" by Bergstra and Bengio: random search is more efficient than grid search
- **Space filling designs:** latin hypercube design, orthogonal arrays, etc.



Grid search



Random search

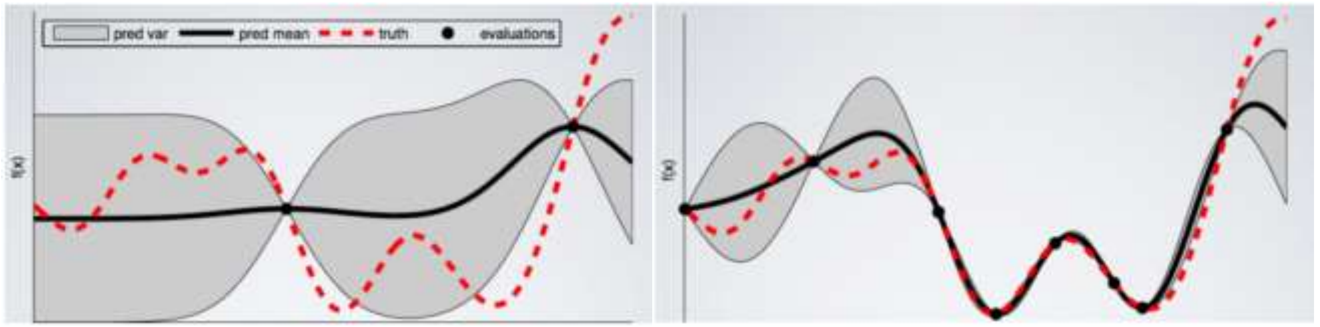


Latin Hyper-cube search

- Grid search is implemented in GridSearchCV in sklearn in python.
- Random search is implemented in RandomizedSearchCV in sklearn in python
- GPyOpt – space filling designs

Sequential search methods: Bayesian optimization

- Grid and random search are completely uninformed by past evaluations. Both approaches may waste time searching area that is unpromising based on past evaluations.
- Bayesian parameter optimization takes past evaluations into consideration. By evaluating hyperparameters that appear more promising from past results, Bayesian methods can find better model settings than random search in fewer iterations.
    1. Build a surrogate probability model on the past evaluation results, $\hat{P}(score|parameter)$. It is a distribution, not a deterministic function in this approach.
    2. Find the hyperparameters that perform best on the surrogate model.
    3. Apply these hyperparameters to build the model and evaluate the scores
    4. Update the surrogate model incorporating the new results
    5. Repeat steps 2–4 until max iterations or time is reached.



Surrogate model with 2 evaluations (left) and 8 evaluations (right)

Summary of search methods

- There are other sequential search methods like TPE (Tree-structured Parzen Estimators), Hyperband, etc.
- All sequential algorithms exploit previous information to do intelligent searches and hence are more efficient than batch techniques.
- The primary advantage of the batch methods is their simplicity.
- There are several open source software packages available for implementing the algorithms discussed above:
  - **Spearmint:** https://github.com/HIPS/Spearmint
  - **RoBO:** http://automl.github.io/RoBO/
  - **GPyOpt:** https://github.com/SheffieldML/GPyOpt
  - **GPflowOpt:** https://github.com/GPflow/GPflowOpt
  - **Hyperband:** https://github.com/zygmuntz/hyperband

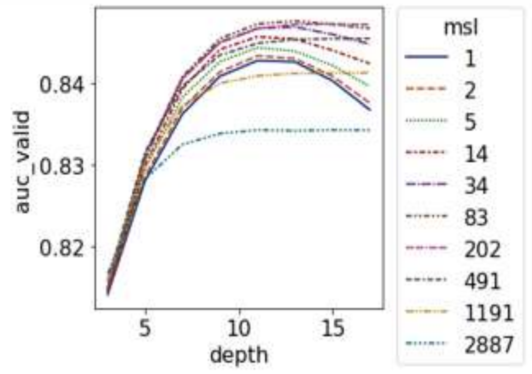
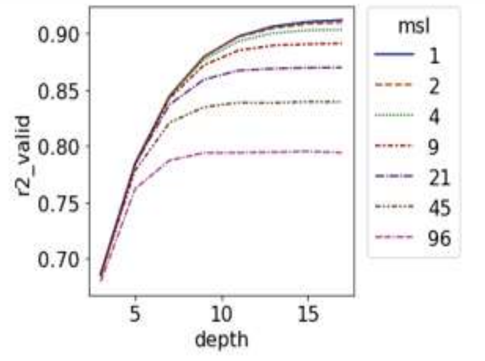
Points to Consider When Evaluating Hyperparameter Optimization

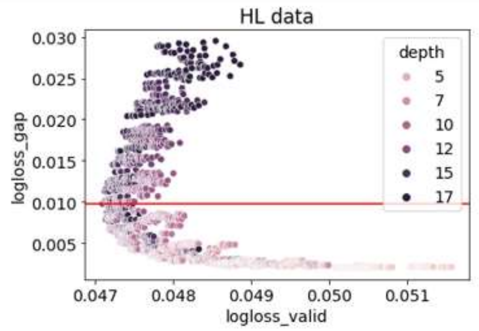
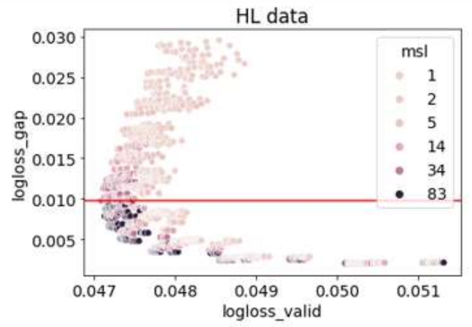
- Were the appropriate hyperparameters considered?
- Was an appropriate search space over the hyperparameters considered?
  Note: Be careful when a value on the boundary of the
- Is the evaluation of the models appropriate?
  - Is the holdout data used appropriate for the intended use?
  - Is the evaluation metric appropriate for the intended use?
  - Should additional factors beyond performance be considered in choosing a model?

Hyper-Parameter Tuning Investigation

Select insights from internal HP study on RF for structured data

- Depth and Minimum samples per leaf (MSL) are the most important parameters
- As long as number of trees > 100, effect of further increase in number of trees is negligible
- Optimal depth > 11
- Low interaction between HPs
  - Exception: Depth vs MSL interaction present.
- Plot in row 1 shows performance (AUC/ R^2) for two pairs of HP for two datasets (Home Lending, Bike Share)
- Plot in row 2 shows high depth can lead to over-fitting, however tuning MSL can control this.
- More on HP tuning later!

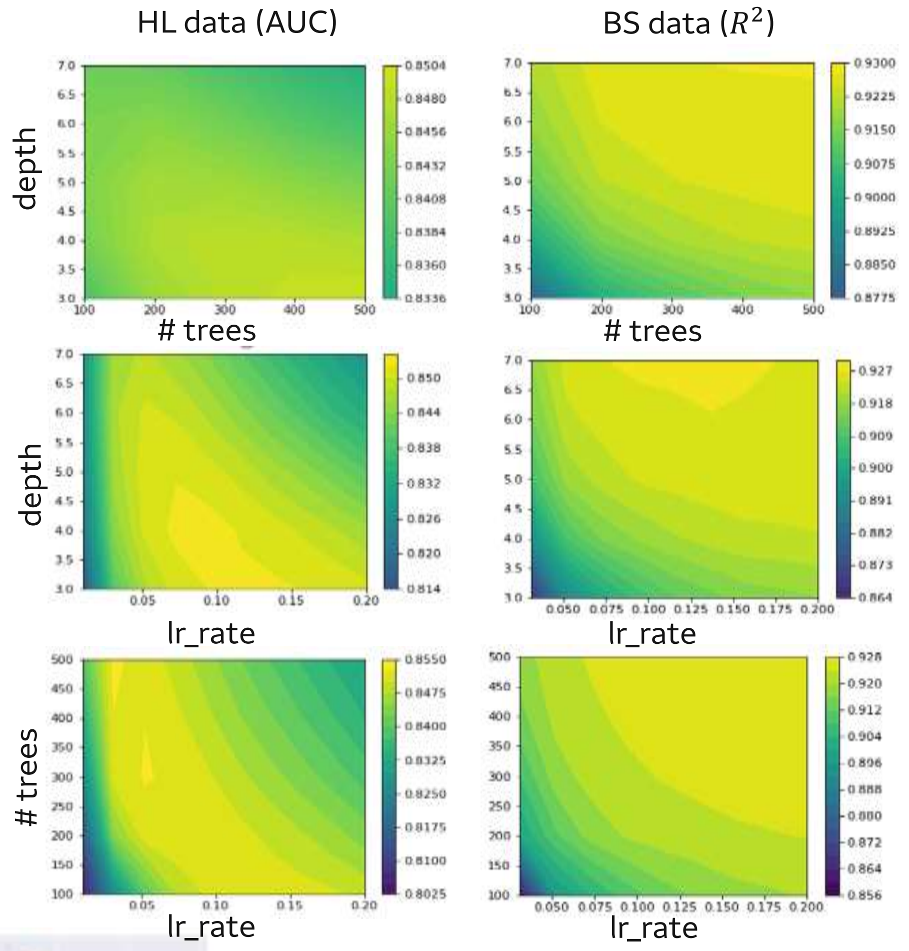
HP pairs	HL	BS
Depth vs MSL		

Datasets	Depth	MSL
HL		

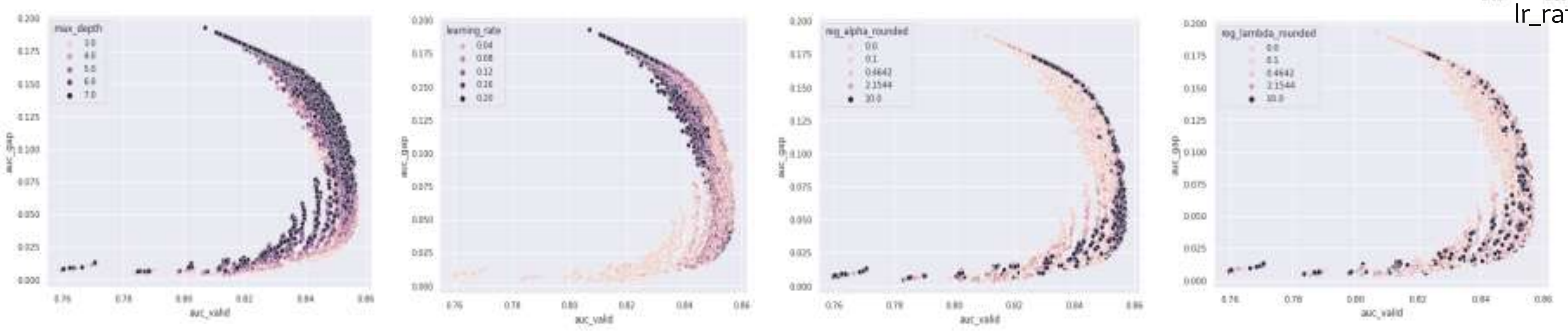
https://arxiv.org/ftp/arxiv/papers/2211/2211.08536.pdf

Select insights from internal HP study on XGBoost for structured data

- HP with highest impact is usually depth.
  - Optimal depth is <8 in contrast with RF
- Trees and Learning rate(lr_rate) are also heavy impact HP with high interaction
  - Optimal trees should be obtained using early stopping
  - If max number of trees is set at 500, lr_rate has to be > 0.01
- L1 penalty has stronger regularization effect compared to L2 of same strength.
- Plots on right show average algorithm performance of two real datasets (Home Lending, Bike Share) for different HP combinations
- Plots below show high depth and lr_rate and lead to over-fitting and applying regularization can reduce over-fitting



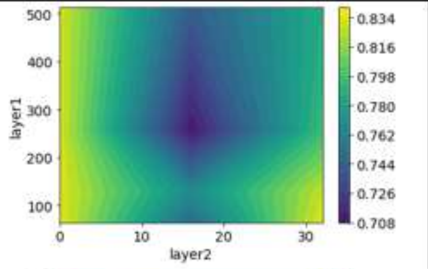
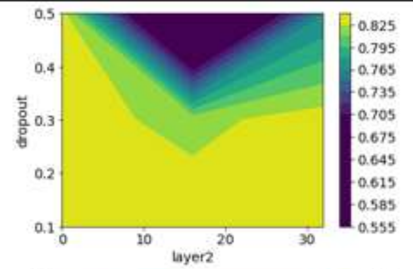
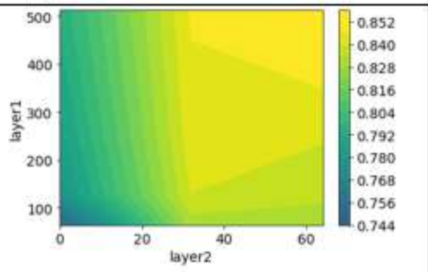
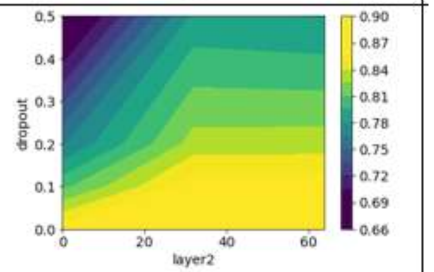
HL data



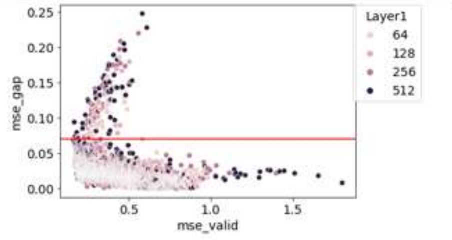
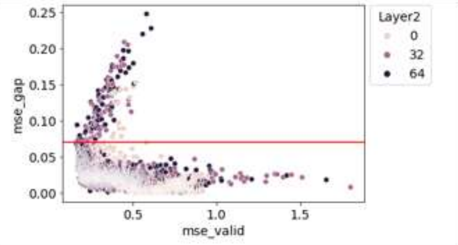
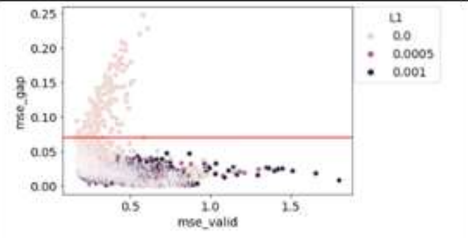
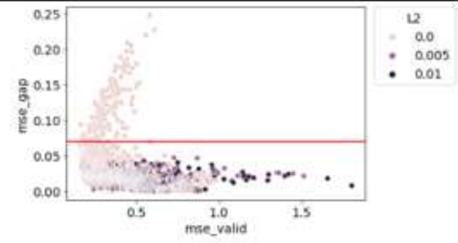
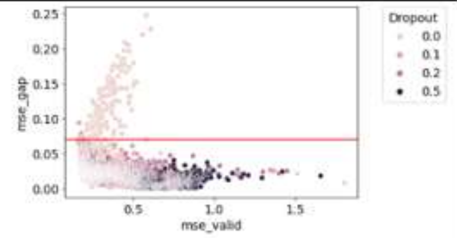
https://arxiv.org/ftp/arxiv/papers/2211/2211.08536.pdf

Select insights from internal HP study on FFNN for structured data

- Learning rate needs to be low <0.001
- L1 has very strong effect → even small L1 penalty (0.01) can reduce performance greatly
- High dropout (>0.3) when applied on small layers (<64) will reduce performance.
- High interaction between HPs
- Plots on right show average algorithm performance of two real datasets (Home Lending and Bike Share) for select HP combinations
- Plots below show larger networks can cause over-fitting and applying regularization can reduce over-fitting, and aggressive regularization reduces model performance.

	Layer1 vs Layer2	Dropout vs Layer2
HL: AUC		
BS: R^2		

BS data

BS					
					

https://arxiv.org/ftp/arxiv/papers/2211/2211.08536.pdf

Adopting two stage approaches for HP tuning in large datasets

- For extremely large datasets (>1M observations and >50 predictors), training each model takes a **long time**.
- If number of HPs are large then **reliable** evaluation of HP space takes **multiple model training**
- A two stage approach can effectively **reduce the search space** if chosen carefully
  - Tune important parameters in stage 1 and hold other HP at default settings
  - Based on optimal HP in reduced space from stage 1, tune other HPs.
  - Loss in accuracy is through two stage approach is low.
- Table from the right shows results from the two stage strategies studied in https://arxiv.org/ftp/arxiv/papers/2211/2211.08536.pdf.
  - Relative decrease in performance by tuning HP models in two stages is low.

Algorithm	Datasets	Metric	Total number of models in full grid search	% of models with performance > two stage_opt model	Relative decrease in performanc e
RF	HL	AUC	3,600	0.17%	0.05%
		Logloss	3,600	0.03%	0.06%
	BS	MSE	1,512	best	0%
XGB	HL	AUC	6,250	best	0%
		Logloss	6,250	best	0%
	BS	MSE	6,250	1.47%	1.9%
FFNN	HL	AUC	5,184	Best	0%
		Logloss	5,184	Best	0%
	BS	MSE	6,912	Best	0%